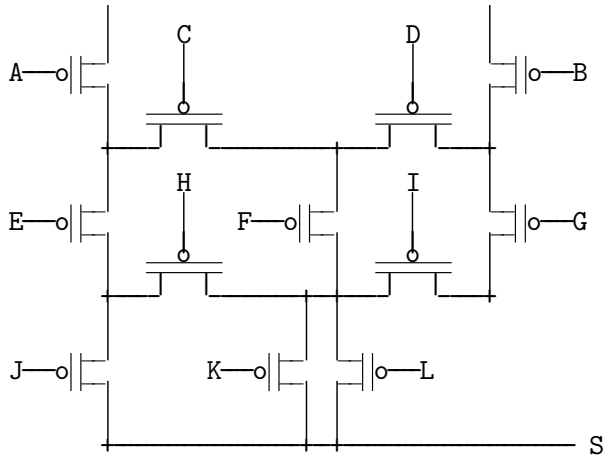
Université de Nanterre vendredi 11 mai 2018
architecture des ordinateurs contrôle

Que valent CF, OF, ZF et SF après les opérations 3+13, 4+8, 3-14, 11-9, 7-10, 2+3, 1-0, 15-7, 8-15, 0-3, 9+9, 6+6, 9-3, 13+13, sur des nombres codés sur 4 bits.

Complétez le bas du schéma avec autant de transistors.




```
f:  loadimm16 r2,1
    xor  r3,r3,r3
l1: sub  r0,r2,r0
    jc   l2
    load r1,r4
    neg  r4,r5
    sub  r4,r5,r6
    movccl r5,r4  // if(  <  ) r4=r5   signé
    add  r3,r4,r3
    add  r1,r2,r1
    jmp  l1
l2: mov  r3,r0
    ret
```

Donnez un équivalent simple en C de la fonction `f`.

Que vaut `x` dans `int t[]={3,6,-1,-5,3,-4,2}, x=f(7,t);`

Donnez une formule donnant la valeur de `f(n,t)`. A défaut dîtes en une phrase ce que calcule `f`.

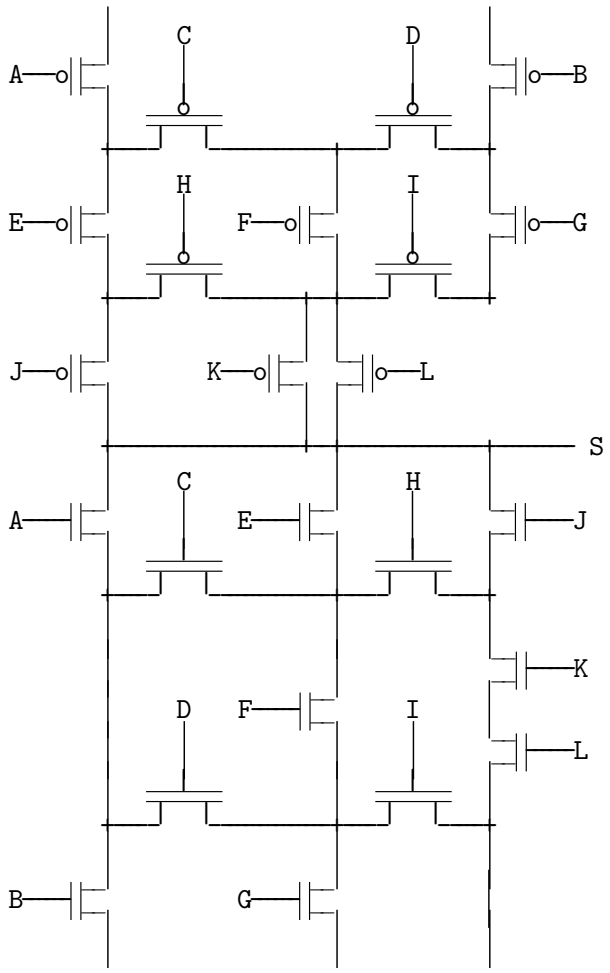
Même question si on remplace `movccl` par `movccg`.

Redonnez la valeur de `x` et la formule de `f(n,t)` si on duplique la ligne `add r3,r4,r3`.

Ecrivez en C puis en assembleur la fonction `int g(int n, int *t);` qui rend $\sum_{i=0}^{n-1} \left|t[i]^2 - 100\right|$.

Corrigé

non signé	signé	CF	OF	ZF	SF
3+ 13= 0	3+ -3= 0	1	0	1	0
4+ 8=12	4+ -8=-4	0	0	0	1
3- 14= 5	3- -2= 5	1	0	0	0
11- 9= 2	-5- -7= 2	0	0	0	0
7- 10=13	7- -6=-3	1	1	0	1
2+ 3= 5	2+ 3= 5	0	0	0	0
1- 0= 1	1- 0= 1	0	0	0	0
15- 7= 8	-1- 7=-8	0	0	0	1
8- 15= 9	-8- -1=-7	1	0	0	1
0- 3=13	0- 3=-3	1	0	0	1
9+ 9= 2	-7+ -7= 2	1	1	0	0
6+ 6=12	6+ 6=-4	0	1	0	1
9- 3= 6	-7- 3= 6	0	1	0	0
13+ 13=10	-3+ -3=-6	1	0	0	1



```
                //          r0    r1    r2   r3   r4       r5   r6
f:  loadimm16 r2,1// int f(int n,int*t) // 1    s    x        y    z
    xor  r3,r3,r3 // { int s=0;         //         *t,|*t|   -*t  x-y
l1: sub  r0,r2,r0
    jc   l2       //   while(n--)
    load r1,r4    //   { int x=*t,
    neg  r4,r5    //         y=-x;      // y=-*t
    sub  r4,r5,r6 //   //int z=x-y;
    movccl r5,r4  //     if(x<y) x=y;   // x=max(x,y)=max(*t,-*t)=|*t|
    add  r3,r4,r3 //     s+=x;          // s+=|*t++|
    add  r1,r2,r1 //     t++;
    jmp  l1       //   }
l2: mov  r3,r0    //   return s;
    ret           // }
```

```
int f(int n,int*t)      int g(int n,int*t)
{ int s=0;              { int s=0;
  while(n--)              while(n--)
  { int x=*t++;           { int x=*t++;    x=x*x-100;
    if(x<-x) x=-x;          if(x<-x) x=-x;
    s+=x;                   s+=x;
  }                       }
  return s;               return s;
}                       }
```

$$x = 3+6+1+5+3+4+2 = 24 \qquad f(n,t) = \sum_{i=0}^{n-1} |t[i]|$$

C'est la somme des valeurs absolues des nombres contenus dans le tableau `t` de dimension `n`.

```
    movccl r5,r4  //     if(x<y) x=y;   // x=max(x,y)=max(*t,-*t)=|*t|
    movccg r5,r4  //     if(x>y) x=y;   // x=min(x,y)=min(*t,-*t)=-|*t|
```

$$x = -3-6-1-5-3-4-2 = -24 \qquad f(n,t) = \sum_{i=0}^{n-1} -|t[i]| = -\sum_{i=0}^{n-1} |t[i]|$$

C'est l'opposé de la somme des valeurs absolues des `n` premiers nombres du tableau `t`.

En dupliquant la ligne `add r3,r4,r3` on a :

$$x = 3+3+6+6+1+1+5+5+3+3+4+4+2+2 = 48 \qquad f(n,t) = \sum_{i=0}^{n-1} 2|t[i]| = 2\sum_{i=0}^{n-1} |t[i]|$$

C'est le double de la somme des valeurs absolues des `n` premiers nombres du tableau `t`.

Dans la fonction `f` donnée dans l'énoncé, on ajoute une ligne après le `loadimm16 r2,1` et deux lignes après le `load r1,r4`.

```
g:  loadimm16 r2,1
    loadimm16 r7,100
    ...
    load r1,r4    //   int x=*t;
    mul  r4,r4,r4 //   x*=x;
    sub  r4,r7,r4 //   x-=100;
```

Barème

1) 4pt

Chaque opération partiellement fausse ou manquante : -0.3pt.

2) 6pt

-1pt pour un transistor dont une patte est mal raccordée.

-1.5pt pour un transistor dont les deux pattes sont mal raccordées.

3) 6pt

`f` en C	1.5 pt-0.5pt par erreur comme argument manquant, test de boucle faux, incrément oublié
24	0.75pt
formule	0.75pt
-24	0.75pt
formule	0.75pt
48	0.75pt
formule	0.75pt

4) 4pt

`g` en C	1 pt
`loadimm16 r7,100`	1 pt
`mul r4,r4,r4 // x*=x;`	1 pt
`sub r4,r7,r4 // x-=100;`	1 pt

-0.5pt pour chaque erreur.